

The image is a large, symmetrical, abstract graphic composed of the letters 'S' and 'Y' arranged in a grid-like pattern. The overall shape is a stylized 'Y' or a complex letter 'H'. The top part is a wide horizontal bar made of 'S's, with 'Y's forming a central vertical column. The sides are also made of 'S's, with 'Y's forming a central vertical column. The bottom part is a wide horizontal bar made of 'S's, with 'Y's forming a central vertical column. The entire graphic is composed of black letters on a white background.

```

LL               IIIII
LL               IIIII
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LL               III
LLLLLLLLLLLLLL  IIIII
LLLLLLLLLLLLLL  IIIII

                SSSSSSSS
                SSSSSSSS
                SS
                SS
                SS
                SS
                SSSSSS
                SSSSSS
                SS
                SS
                SS
                SS
                SSSSSSSS
                SSSSSSSS

```

(1) 42  
(2) 57  
(3) 120  
(5) 182  
(6) 323  
(10) 425  
(12) 532

HISTORY ; DETAILED  
DECLARATIONS  
ALLOCSWPAREA - ALLOCATE A SWAP AREA IN A PAGE FILE  
ALLOCPAGFIL - ALLOCATE A PAGING FILE SPACE  
ALLOCPAGFIL - ALLOCATE A PAGING FILE SPACE  
DALCPAGFIL - DEALLOCATE PAGE IN PAGING FILE  
ALC\_PGFLVBN Allocate specific blocks in paging file



```
0000 1 .TITLE PAGEFILE - ALLOCATE / DEALLOCATE PAGING FILE
0000 2 .IDENT 'V04-000'
0000 3
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26
0000 27 ++
0000 28 FACILITY: EXECUTIVE, MEMORY MANAGEMENT SUBROUTINES
0000 29
0000 30 ABSTRACT:
0000 31
0000 32 THIS MODULE CONTAINS THE ROUTINES FOR ALLOCATING AND DEALLOCATING
0000 33 PAGES FROM A PAGING FILE.
0000 34
0000 35 ENVIRONMENT:
0000 36
0000 37 THESE ROUTINES RUN IN KERNEL MODE AND MUST BE CALLED WITH
0000 38 IPL AT SYNCH OR HIGHER.
0000 39
0000 40 --
0000 41
0000 42 .SBTTL HISTORY ; DETAILED
0000 43
0000 44 AUTHOR: PETER H. LIPMAN , CREATION DATE: 29-OCT-76
0000 45
0000 46 MODIFIED BY:
0000 47
0000 48 V03-004 WMC00001 Wayne Cardoza 09-Jul-1984
0000 49 Make the pagefile full messages more accurate.
0000 50
0000 51 V03-003 KDM0002 Kathleen D. Morse 28-Jun-1982
0000 52 Add $DYNDEF.
0000 53
0000 54 **
0000 55
```

```
0000 57 .SBTTL DECLARATIONS
0000 58
0000 59 :
0000 60 : INCLUDE FILES:
0000 61 :
0000 62 $DYNDEF ;DYNAMIC DATA STRUCTURE TYPE DEFINITIONS
0000 63 $PFLDEF ;PAGE FILE CONTROL BLOCK DEFINITIONS
0000 64 $PTEDEF ;PAGE TABLE ENTRY DEFINITIONS
0000 65 $RSNDEF ;RESOURCE NAME DEFINITIONS
0000 66 :
0000 67 : EXTERNAL SYMBOLS:
0000 68 :
0000 69 :
0000 70 :
0000 71 : MACROS:
0000 72 :
0000 73 :
0000 74 :
0000 75 : EQUATED SYMBOLS:
0000 76 :
0000 77 :
0000 78 :
0000 79 : OWN STORAGE:
0000 80 :
0000 81 :
00000000 82 .PSECT $$$220, LONG ; SWAPPER/SCHEDULER DATA
0000 83 .ALIGN LONG ;
0000 84
0000 85 MMG$GL_NULLPFL:: ; NULL PFL SERVES AS PLACEHOLDER
00000000 0000 86 .LONG 0 ; BITMAP POINTER, 0 IF TABLE NOT IN USE
00000000 0004 87 .LONG 0 ; ADDRESS OF MPW_WRTCLUSTER SIZE AREA
0024 0008 88 .WORD PFL$C_LENGTH
23 000A 89 .BYTE DYN$C_PFL
00 000B 90 .BYTE 0
00000000 000C 91 .LONG 0 ; PAGE FAULT CLUSTER
00000000 0010 92 .LONG 0 ; WINDOW POINTER, *** FILLED IN BY INIT
00000000 0014 93 .LONG 0 ; VBN, *** FILLED IN BY INIT
00000000 0018 94 .LONG 0 ; BITMAP SIZE
003FFFFFF 001C 95 .LONG 0 ; FREE PAGE COUNT IN THIS FILE
00000000 0020 96 .LONG PTE$M_PGFLVB ; PAGE FILE VBN MASK
0024 97 .LONG 0 ; ACCOUNT FOR EXTENDED LENGTH
0024 98 :
0024 99 : POINTER TO VECTOR OF PAGE/SWAP FILE CONTROL BLOCKS
0024 100 :
00000000 0024 101 MMG$GL_PAGSWPVC::
0028 102 .LONG 0
0028 103 :
0028 104 :
0028 105 : MAXIMUM PAGE FILE INDEX CURRENTLY IN USE
0028 106 :
0028 107 :
00000000 0028 108 MMG$GL_MAXPFIDX::
0028 109 .LONG 0
002C 110 :
002C 111 MMG$GW_MINPFIDX::
002C 112 SGN$GW_SWPFILCT:: ; Count of swapfile slots
0000 002C 113 .WORD 0
```

```
002E 114  
002E 115 :  
002E 116 : Most of the routines in this module are permanently resident  
002E 117 :  
00000000 118 .PSECT $MMGCOD
```



```
0000 120 .SBTTL ALLOCSWPAREA - ALLOCATE A SWAP AREA IN A PAGE FILE
0000 121
0000 122 :++
0000 123 : FUNCTIONAL DESCRIPTION:
0000 124 :
0000 125 : THIS ROUTINE ALLOCATES A CLUSTER OF PAGES FROM THE SPECIFIED PAGE FILE.
0000 126 :
0000 127 : CALLING SEQUENCE:
0000 128 :
0000 129 : BSBW MMG$ALLOCSWPAREA
0000 130 :
0000 131 : INPUT PARAMETERS:
0000 132 :
0000 133 : r0 = VBN in paging file representing start of current allocation
0000 134 : r1 = current allocation size
0000 135 : r2 = new request size
0000 136 :
0000 137 : IMPLICIT INPUTS:
0000 138 :
0000 139 : none
0000 140 :
0000 141 : OUTPUT PARAMETERS:
0000 142 :
0000 143 : r0 = page file vbn (greater than 0) if successful
0000 144 : r2 = number of pages allocated
0000 145 : r1,r3 destroyed
0000 146 :
0000 147 : IMPLICIT OUTPUTS:
0000 148 :
0000 149 : none
0000 150 :
0000 151 : COMPLETION CODES:
0000 152 :
0000 153 : positive condition code indicates success
0000 154 : negative condition code indicates failure
0000 155 : zero condition code indicates failure because request too early in boot
0000 156 :
0000 157 : SIDE EFFECTS:
0000 158 :
0000 159 : none
0000 160 :
0000 161 :--
```

```

30      0000 163 mmg$allocswparea::
55      BB 0000 164      pushr    #^m<r4,r5>      ; save work registers
01      D4 0002 165      clr     r5              ; indicator for no paging files at all
53      54 01      D0 0004 166      movl    #1,r4      ; start scan at file index 1
10      0024'DF44 D0 0007 167 10$:  movl    @w^mmg$gl_pagswpvc[r4],r3 ; get address of next page file block
ED 54    10 23 A3 00      E0 000D 168      bbs     #pfl$y_initd,pfl$b_flags(r3),30$ ;branch if this one initd
00000028'GF F3 0012 169 20$:  aobleq   q^mmg$gl_maxpfidx,r4,10$ ; loop through all page files
52      01      CE 001A 170      mnegl   #1,r2      ; assume unlimited growth size
50      55      D0 001D 171      movl    r5,r0      ; set flag indicating if we are booting
13      11 0020 172      brb     40$             ; use common exit
55      D7 0022 173 30$:  decl     r5              ; indicate valid paging file exists
OFF0 8F BB 0024 174      pushr    #^m<r4,r5,r6,r7,r8,r9,r10,r11> ;save volatile registers
OE      10 0028 175      bsbb     mmg$allocpagfil1 ; allocate new area in page file
OFF0 8F BA 002A 176      popr     #^m<r4,r5,r6,r7,r8,r9,r10,r11> ;restore volatile registers
E2      13 002E 177      beql     20$             ; try next page file
50      08 18 54      F0 0030 178      insv   r4,#24,#8,r0 ; save swap file index
30      BA 0035 179 40$:  popr     #^m<r4,r5>      ; restore registers
05      05 0037 180      rsb      ; return to caller
```



```

0038 182      .SBTTL ALLOCPAGFIL - ALLOCATE A PAGING FILE SPACE
0038 183
0038 184      :++
0038 185      : FUNCTIONAL DESCRIPTION:
0038 186      :
0038 187      :     THIS ROUTINE ALLOCATES A CLUSTER OF PAGES FROM THE SPECIFIED PAGE FILE.
0038 188      :
0038 189      : CALLING SEQUENCE:
0038 190      :
0038 191      :     BSBW      MMG$ALLOCPAGFIL1
0038 192      :
0038 193      : INPUT PARAMETERS:
0038 194      :
0038 195      :     r0 = VBN in paging file representing start of current allocation
0038 196      :     r1 = current allocation size
0038 197      :     r2 = new request size
0038 198      :     r3 = page file index
0038 199      :
0038 200      : IMPLICIT INPUTS:
0038 201      :
0038 202      :     NONE
0038 203      :
0038 204      : OUTPUT PARAMETERS:
0038 205      :
0038 206      :     R0 = PAGE FILE VBN (GREATER THAN 0) IF SUCCESSFUL
0038 207      :     R2 = NUMBER OF PAGES ALLOCATED
0038 208      :
0038 209      : IMPLICIT OUTPUTS:
0038 210      :
0038 211      :     NONE
0038 212      :
0038 213      : COMPLETION CODES:
0038 214      :
0038 215      :     Z-BIT SET IF FAILURE
0038 216      :     Z-BIT 0 IF SUCCESS
0038 217      :
0038 218      : SIDE EFFECTS:
0038 219      :
0038 220      :     MMG$ALLOCPAGFIL2 has register content dependencies on this routine!
0038 221      :
0038 222      :     This routine depends on allocation sizes to be multiples of 8 for
0038 223      :     reasonable search times now that this is first fit. This implies
0038 224      :     that the modified page writer cluster size should be equal to
0038 225      :     the swap space allocation increment, to allow the local "memory"
0038 226      :     to work reasonably. Also the minimum modified page writer cluster size
0038 227      :     should be at least 16 blocks for correct resource failure continuation,
0038 228      :     this allows some emergency 8 byte blocks to be allocated.
0038 229      :
0038 230      :--

```

```

56 63 D0 0038 232 MMGSALLOCPAGFIL1:
57 14 A3 D0 0038 233      movl    pfl$$_bitmap(r3),r6      ;address of start of map
5A 57 56 C1 003F 234      movl    pfl$$_bitmapsiz(r3),r7    ;number of bytes in map
5B 01 CE 0043 235      addl3    r6,r7,r10      ;get end of map address
58 52 FD 8F 78 0046 236      mnegl    #1,r11      ;materialize a minus for use
59 50 08 18 EF 0048 237 20$:      pushr    #^m<r0,r1,r2,r3>      ;save the inputs, (r3 is now address)
53 0024'DF49 D1 0052 238      ashl     #-3,r2,r8      ;make size into byte count
50 50 18 00 EF 004D 239      extzv    #24,#8,r0,r9      ;get the page file index
51 51 50 40 13 0052 240      cmpl     @w^mmg$gl_pagswpc[r9],r3 ;is this in the same page file?
59 59 FD 8F 78 0058 241      bneq     60$      ;branch if not, try for simple allocate
54 51 FD 8F 78 005A 242      extzv    #0,#24,r0,r0      ;get the input VBN
55 58 54 C3 0061 243      beql     60$      ;branch if not holding current space
59 59 FD 8F 78 0065 244      addl3    r0,r1,r9      ;get ending block
54 51 FD 8F 78 006A 245      ashl     #-3,r9,r9      ;get byte offset of area after this one
55 58 54 C3 006A 246      ashl     #-3,r1,r4      ;r0+r1 always yield (multiple of 8)+1
59 59 FD 8F 78 006F 247      subl3    r4,r8,r5      ;current size in groups of 8
54 51 FD 8F 78 0073 248      bgtr     30$      ;number of additional needed blocks
55 58 54 C3 0075 249      bsbw     mmg$deallocpagfil ;branch if this is an expansion
59 59 FD 8F 78 0078 250      popr     #^m<r0,r1,r2,r3> ;free current holding if contraction
54 51 FD 8F 78 007A 251      clrq     r0      ;restore regs
55 58 54 C3 007C 252      brb     20$      ;indicate holding freed
59 59 FD 8F 78 007E 253      :
54 51 FD 8F 78 007E 254      : The end of map condition is handled by having a non-allocatable byte at
55 58 54 C3 007E 255      : the end of the map. This allows the skpc to failure terminate.
59 59 FD 8F 78 007E 256      :
54 51 FD 8F 78 007E 257      :
6649 55 5B 3B 007E 258 30$:      skpc     r11,r5,(r6)[r9]      ;find additional contiguous free space
6649 55 00 61 00 2C 0083 259      bneq     60$      ;branch if non-free blocks in area
54 51 FD 8F 78 0085 260      movc5    #0,(r1),#0,r5,(r6)[r9] ;mark these blocks allocated
55 58 54 C3 008C 261      :
59 59 FD 8F 78 008C 262      : It is safe not to update STARTBYTE down this path since this is an allocate.
54 51 FD 8F 78 008C 263      : This is also probably desirable to lessen start of map searches.
55 58 54 C3 008C 264      :
59 59 FD 8F 78 008C 265      popr     #^m<r0,r1,r2,r3> ;restore regs
54 51 FD 8F 78 008E 266      subl     r2,r1      ;note input VBN is output VBN
59 59 FD 8F 78 0091 267      addl     r1,pfl$$_frepagcnt(r3) ;get additionally allocated blocks
54 51 FD 8F 78 0091 268      bicpsw   #4      ;(count is negative)
55 58 54 C3 0095 269      rsb      ;update count of available pages
59 59 FD 8F 78 0097 270      :
54 51 FD 8F 78 0098 271      : allocation failure return
55 58 54 C3 0098 272      :
59 59 FD 8F 78 0098 273      :
54 51 FD 8F 78 0098 274      :
23 A3 04 BA 0098 275 40$:      popr     #^m<r0,r1,r2,r3> ;restore regs
55 58 54 C3 009A 276      bisb     #pfl$m_swpcfilful,pfl$b_ ;set flag indicating file full
59 59 FD 8F 78 009E 277      bispsw   #4      ;indicate failure, no deallocation!
54 51 FD 8F 78 00A0 278      rsb      ;z-bit set
55 58 54 C3 00A1 279      :
59 59 FD 8F 78 00A1 280      : new allocation
54 51 FD 8F 78 00A1 281      :
55 58 54 C3 00A1 282 60$:      movzwl   r11,r5      ;set up for 65536 byte locate
59 59 FD 8F 78 00A4 283      cmpb     r2,pfl$b_allocsiz(r3) ;is this standard request size?
54 51 FD 8F 78 00AB 284      blss     70$      ;branch if not, search from start
55 58 54 C3 00AA 285      movl     pfl$$_startbyte(r3),r1 ;set up to start from first known free
59 59 FD 8F 78 00AE 286      bneq     80$      ;branch if we know where
54 51 FD 8F 78 00B0 287 70$:      movl     r6,r1      ;set up to scan map from start
55 58 54 C3 00B3 288
```

```
57 5A 51 C3 00B3 289 80$: subl3 r1,r10,r7 ;calc number of bytes remaining to scan
    DF 13 00B7 290 beql 40$ ;branch if at end of map
    55 57 D1 00B9 291 cmpl r7,r5 ;less than 65536 bytes to scan?
    03 18 00BC 292 bgeq 90$ ;branch if not
    55 57 D0 00BE 293 movl r7,r5 ;set scan amount to what's left
61 55 5B 3A 00C1 294 90$: locc r11,r5,(r1) ;find a byte aligned area with 8 blocks
    EC 13 00C5 295 beql 80$ ;branch if no free clusters in area
    00C7 296
    00C7 297 : The end of map condition is handled by having a non-allocatable byte at
    00C7 298 : the end of the map. This allows the skpc to failure terminate.
    00C7 299
    61 58 5B 38 00C7 300 skpc r11,r8,(r1) ;is this sequence long enough?
    E6 12 00CB 301 bneq 80$ ;branch if not, look for another
    51 58 C2 00CD 302 subl r8,r1 ;get back start address of field
61 58 00 61 00 2C 00D0 303 movc5 #0,(r1),#0,r8,(r1) ;allocate area, preserve r1 address
    57 51 56 C3 00D6 304 subl3 r6,r1,r7 ;save start byte to return it
    59 53 D0 00DA 305 movl r3,r9 ;save address of end of this area
    03 BA 00DD 306 popr #^m<r0,r1> ;restore regs for deallocations, if any
53 50 08 18 EF 00DF 307 extzv #24,#8,r0,r3 ;get the page file index
50 50 18 00 EF 00E4 308 extzv #0,#24,r0,r0 ;get the input VBN
    09 13 00E9 309 beql 95$ ;branch if no previous holding
53 0024'DF43 D0 00EB 310 movl @w^mmg$gl_pagswpyc[r3],r3 ;get page file control block address
    014F 30 00F1 311 bsbw mmg$deallocpagfil ;free up the space
    0C BA 00F4 312 95$: popr #^m<r2,r3> ;restore the request size, PFL addr
    22 A3 52 91 00F6 313 cmpb r2,pfl$b_allocsiz(r3) ;was this for current request size
    04 A3 59 D0 00FA 314 bneq 100$ ;branch if not, don't affect memory
    18 A3 52 C2 0100 315 movl r9,pfl$l_startbyte(r3) ;update memory for future reference
50 57 03 78 0104 316 100$: subl r2,pfl$l_frepagcnt(r3) ;update count of available pages
    50 D6 0108 317 ashl #3,r7,r0 ;multiply byte number*8 to get VBN
    05 010A 318 incl r0 ;VBN's need to be based at 1
    010B 319 rsb ;return, z-bit=0
    010B 320
    010B 321 BADALLOC:
    010B 322 BUG CHECK BADPAGFILA,FATAL ;BAD PAGE FILE ADDRESS ALLOCATED
    010F 323 .SBTTL ALLOCPAGFIL - ALLOCATE A PAGING FILE SPACE
```



```

010F 325 :++
010F 326 : FUNCTIONAL DESCRIPTION:
010F 327 :
010F 328 :     THIS ROUTINE ALLOCATES THE FIRST CONTIGOUS SET OF BLOCKS FROM
010F 329 :     THE SPECIFIED PAGE FILE.
010F 330 :
010F 331 : CALLING SEQUENCE:
010F 332 :
010F 333 :     BSBW    MMGSALLOCPAGFIL2      ; must occur just after a call
010F 334 :                                     ; to MMGSALLOCPAGFIL
010F 335 :
010F 336 : INPUT PARAMETERS:
010F 337 :
010F 338 :     r3 = page file control block address
010F 339 :     r6 = address of start of bitmap
010F 340 :     r10 = end address of bitmap
010F 341 :     r11 = 65536 (maximum size for a string instruction length)
010F 342 :
010F 343 : IMPLICIT INPUTS:
010F 344 :
010F 345 :     NONE
010F 346 :
010F 347 : OUTPUT PARAMETERS:
010F 348 :
010F 349 :     R0 = PAGE FILE VBN (GREATER THAN 0) IF SUCCESSFUL
010F 350 :     R2 = NUMBER OF PAGES ALLOCATED
010F 351 :
010F 352 : IMPLICIT OUTPUTS:
010F 353 :
010F 354 :     NONE
010F 355 :
010F 356 : COMPLETION CODES:
010F 357 :
010F 358 :     Z-BIT SET IF FAILURE
010F 359 :     Z-BIT 0 IF SUCCESS
010F 360 :
010F 361 : SIDE EFFECTS:
010F 362 :
010F 363 :     none
010F 364 :--

```

Address	Disassembly	Comment
00000045	010F	366 fragmsg:
00000117	010F	367 .long 20\$-10\$
	0113	368 .long 10\$
	0117	369 10\$: .ascii <13><10>-
2D 57 2D 4D 45 54 53 59 53 25 0A 0D	0117	370
61 50 20 2C 47 41 52 46 45 47 41 50	0123	
6C 64 61 62 20 65 6C 69 66 20 65 67	012F	
64 65 74 6E 65 6D 67 61 72 66 20 79	013B	
6E 6F 63 20 6D 65 74 73 79 73 20 2C	0147	
	0153	
	015A	371 \SYSTEM-W-PAGEFRAG, Page file badly fragmented, system continuing\-
0A 0D	015A	372 <13><10>
	015C	373 20\$:
	015C	374 critmsg:
0000004B	015C	375 .long 40\$-30\$
00000164	0160	376 .long 30\$
	0164	377 30\$: .ascii <13><10>-
	0165	378 \SYSTEM-W-PAGECRIT, Page file space critical, system trying to continue\-
50 2D 57 2D 4D 45 54 53 59 53 25 0A	0171	
67 61 50 20 2C 54 49 52 43 45 47 41	017D	
65 63 61 70 73 20 65 6C 69 66 20 65	0189	
73 20 2C 6C 61 63 69 74 69 72 63 20	0195	
67 6E 69 79 72 74 20 6D 65 74 73 79	01A1	
65 75 6E 69 74 6E 6F 63 20 6F 74 20	01AD	
	01AF	379 <13><10>
		380 40\$:

```

55 5B 3C 01AF 382 MMG$ALLOCAPGFI2::
51 56 D0 01B2 383      movzwl  r11,r5      ;set up for 65536 byte locate
                                384      movl    r6,r1      ;set up to scan map from start
                                385
57 5A 51 C3 01B5 386 10$:      subl3    r1,r10,r7      ;calc number of bytes remaining to scan
                                387      beql    50$      ;branch if at end of map
55 57 D1 01B8 388      cmpl    r7,r5      ;less than 65536 bytes to scan?
                                389      bgeq    20$      ;branch if not
55 57 D0 01C0 390      movl    r7,r5      ;set scan amount to what's left
61 55 00 3B 01C3 391 20$:      skpc     #0,r5,(r1)      ;find any free blocks
50 61 08 00 EA 01C9 392      beql    10$      ;branch if no free clusters in area
52 50 01 C3 01CE 393      ffs      #0,#8,(r1),r0      ;find the free block
                                394      subl3    #1,r0,r2      ;save start offset
                                395 30$:      incl    r2      ;account for block
FA 61 52 E4 01D4 396      bbsc     r2,(r1),30$      ;loop through contiguous portion of map
52 50 50 C2 01D8 397      subl    r0,r2      ;set r2 number of blocks allocated
18 A3 52 C2 01DB 398      subl    r2,pfl$l_frepagecnt(r3) ;update count of available pages
51 51 56 C2 01DF 399      subl    r6,r1      ;get byte number of free blocks
50 01 A0 7E 01E2 400      movaq    1(r0)[r1],r0      ;form 8*byte number + bit number + 1
51 14 A3 01 78 01E7 401      ashl    #1,pfl$l_bitmapsiz(r3),r1 ;find 1/4 point of VBN's in bitmap
51 50 D1 01EC 402      cmpl    r0,r1      ;is this allocation past 1/4 point?
OB 0000'CF 00' 1F 01EF 403      blssu   50$      ;branch if not, no message needed yet
                                404      bbss     s^exe$vgpflfrag,w^exe$gl_flags,40$ ;branch if reported
51 FF12 CF 07 BB 01F7 405      pushr   #^m<r0,r1,r2>      ;save registers
                                406      movq     fragmsg,r1      ;set up message to output
                                407      bsbb     sendmsg      ;output the message
54 51 51 C1 0200 408      popr     #^m<r0,r1,r2>      ;restore registers
54 51 51 C0 0202 409 40$:      addl3    r1,r1,r4      ;find 3/4 mark in file
54 50 D1 0206 410      addl    r1,r4      ;now have 3/4 VBN
                                411      cmpl    r0,r4      ;is this allocation past 3/4 point
                                412      blssu   50$      ;branch if not
OD 0000'CF 00' 1F 020C 413      bbss     s^exe$vgpflcrit,w^exe$gl_flags,50$ ;branch if reported
51 FF42 CF 07 BB 0214 414      pushr   #^m<r0,r1,r2>      ;save registers
                                415      movq     critmsg,r1      ;set up message to output
                                416      bsbb     sendmsg      ;output the message
                                417      popr     #^m<r0,r1,r2>      ;restore registers
                                418      bicpsw   #4      ;indicate success
                                419 50$:      rsb      ;return, z-bit=0 success, else failure
                                420
                                421 sendmsg:
55 0000'CF 9E 0222 422      movab    w^opa$ucb0,r5      ;set console terminal for broadcast
FDD6' 31 0227 423      brw      ioc$broadcast      ;assume message will get to console
```



```

022A 425 .SBTTL DALCPAGFIL - DEALLOCATE PAGE IN PAGING FILE
022A 426
022A 427 :++
022A 428 : FUNCTIONAL DESCRIPTION:
022A 429 :
022A 430 : THIS ROUTINE DEALLOCATES A SPECIFIED PAGE IN THE SPECIFIED
022A 431 : PAGING FILE.
022A 432 :
022A 433 : CALLING SEQUENCE:
022A 434 :
022A 435 : BSBW MMGSDALCPAGFIL
022A 436 :
022A 437 : INPUT PARAMETERS:
022A 438 :
022A 439 : R0 = PAGE FILE VBN TO DEALLOCATE
022A 440 : R3 = PAGE FILE INDEX
022A 441 :
022A 442 : IMPLICIT INPUTS:
022A 443 : NONE
022A 444 :
022A 445 : OUTPUT PARAMETERS:
022A 446 : R0,R1,R2 DESTROYED
022A 447 : R3 = ADDRESS OF PAGE FILE CONTROL BLOCK
022A 448 :
022A 449 : IMPLICIT OUTPUTS:
022A 450 :
022A 451 : IF THE SPECIFIED PAGING FILE BECOMES NON-EMPTY, THE RESOURCE
022A 452 : AVAILABLE SIGNAL IS ISSUED FOR THE RSN$_PGFILE RESOURCE
022A 453 :
022A 454 : COMPLETION CODES:
022A 455 : NONE
022A 456 :
022A 457 : SIDE EFFECTS:
022A 458 : NONE
022A 459 :
022A 460 :--

```

```
022A 462 .ENABLE lsb
022A 463
022A 464 5$: ;check for checkpoint bit
03 08 50 15 E1 022A 465 bbc #pte$y_chkpt,r0,10$ ;checkpoint bit set?
03 1C A3 15 EO 022E 466 bbs #pte$y_chkpt,pfl$1_maxvbn(r3),10$ ;branch if not a small file
03 03 BA 0233 467 popr #m<r0,r1> ;clean up
05 05 0235 468 rsb ;ignore the deallocation request
0236 469
0236 470 10$: BUG_CHECK BADPAGFIL,FATAL ;BAD PAGE FILE ADDRESS DEALLOCATED
023A 471
023A 472 ; r0 = VBN of block to return
023A 473 ; r3 = page file index
023A 474
023A 475 MMGSDALCPAGFIL::
53 0024'DF43 D0 023A 476 movl @w^mmg$gl_pagswpc[r3],r3 ;get page file control block address
51 01 D0 0240 477 movl #1,r1 ;set count to 1
0243 478 ;fall through
0243 479 ; r0 = VBN of start block to return
0243 480 ; r1 = count
0243 481 ; r3 = address of page file control block
0243 482
0243 483 MMG$DEALLOCPAGFIL::
50 D7 0243 484 decl r0 ;get VBN to base 0
EF 19 0245 485 blss 10$ ;branch if VBN passed was 0
03 BB 0247 486 pushr #m<r0,r1> ;save for later
52 51 50 C1 0249 487 addl3 r0,r1,r2 ;high mark for deallocation
52 52 FD 8F 78 024F 488 decl r2 ;account for count in 0 origin
14 A3 52 C1 0254 489 ashl #-3,r2,r2 ;byte # in map
52 20 D0 0258 490 cmpl r2,pfl$1_bitmapsiz(r3) ;legal page file VBN?
51 52 D1 025A 491 bgequ 5$ ;branch if illegal
03 15 0260 492 movl #32,r2 ;max number single insv can set
52 51 D0 0262 493 30$: cmpl r2,r1 ;free more than 32?
52 50 EC 0265 494 bleq 40$ ;branch if yes
00 00 B3 52 50 0266 495 movl r1,r2 ;set max number to free
00 B3 52 50 FF 0268 496 40$: cmpv r0,r2,apfl$1_bitmap(r3),#0 ;temp check for safety
52 50 FF 026D 497 bneq 10$ ;bugcheck if any of these bit set
52 50 FF 0270 498 insv #-1,r0,r2,apfl$1_bitmap(r3) ;set the bits
18 A3 52 C0 0277 499 addl r2,r0 ;update to next VBN sequence
51 52 C0 027A 500 addl r2,pfl$1_frepagecnt(r3) ;count free pages
51 52 C2 027E 501 subl r2,r1 ;number of blocks to still free
DA 12 0281 502 bneq 30$ ;loop through entire set
03 BA 0283 503 popr #m<r0,r1> ;get back VBN and free count
50 50 FD 8F 78 0285 504 ashl #-3,r0,r0 ;set up to check for 8 block unit freed
51 51 FD 8F 78 028A 505 addl #14,r1 ;round count for worst case crossing
00 B340 51 51 FF 8F 3A 028D 506 ashl #-3,r1,r1 ;number of bytes to check
51 51 FF 8F 3A 0292 507 locc #-1,r1,apfl$1_bitmap(r3)[r0] ;any whole cluster become free?
04 A3 51 D1 0299 508 beql 60$ ;branch if not
50 38 1A 029F 509 cmpl r1,pfl$1_startbyte(r3) ;is freed cluster earlier in map?
50 71 92 02A1 510 bgtru 60$ ;branch if not, note bgtru not bgequ
52 51 01 C1 02A4 511 50$: mcomb -(r1),r0 ;find start byte of free area
51 51 22 A3 9A 02A6 512 beql 50$ ;loop
51 51 FD 8F 78 02AA 513 addl3 #1,r1,r2 ;set start of area
62 51 FF 8F 3B 02AE 514 movzbl pfl$1_allocsiz(r3),r1 ;get current cluster size for this file
04 A3 52 D0 02B3 515 ashl #-3,r1,r1 ;get it in bytes rather than blocks
1F 12 02B8 516 skpc #-1,r1,(r2) ;does this area qualify?
02BA 517 bneq 60$ ;branch if not
movl r2,pfl$1_startbyte(r3) ;save new starting pointer
```

```
0000'CF 22 A3 91 02BE 519 cmpb pfl$b_allocsiz(r3),w^mpw$gw_mwpfc ;are we at maximum size
                                02C4 520 ;we should ever try allocations for?
                                02C4 521 ;branch if at maximum
                                04 13 02C4 521 beql 55$
                                08 80 02C6 522 addb #8,pfl$b_allocsiz(r3) ;try next higher size next time
OA 22 A3 08 80 02C6 522 55$: bbcc #pfl$y_swpfilful,pfl$b_flags(r3),60$ ;branch if not transition
23 A3 02 E5 02CA 523 55$: pushr #^m<r3> ;save pfl address
                                08 BB 02CF 524 ;set up to return swap file available
50 0A DO 02D1 525 movl #rsn$swpfile,r0 ;signal resource available
FD29' 30 02D4 526 bsbw sch$raavail ;restore pfl address
08 05 BA 02D7 527 60$: popr #^m<r3> ;return
                                02D9 528
                                02DA 529
                                02DA 530 .DISABLE lsb
```



```
02DA 532      .SBTTL  ALC_PGFLVBN      Allocate specific blocks in paging file
02DA 533
02DA 534      :++
02DA 535      : FUNCTIONAL DESCRIPTION:
02DA 536      :
02DA 537      :       This routine allocates a specific set of blocks in a paging file
02DA 538      :
02DA 539      : CALLING SEQUENCE:
02DA 540      :
02DA 541      :       BSBW      MMG$ALC_PGFLVBN
02DA 542      :
02DA 543      : INPUT PARAMETERS:
02DA 544      :
02DA 545      :       R0 = VBN of first block to be allocated
02DA 546      :       R1 = Page file index
02DA 547      :       R2 = Number of consecutive blocks to be allocated
02DA 548      :
02DA 549      : IMPLICIT INPUTS:
02DA 550      :       none
02DA 551      :
02DA 552      : OUTPUT PARAMETERS:
02DA 553      :       none
02DA 554      :
02DA 555      : IMPLICIT OUTPUTS:
02DA 556      :       none
02DA 557      :
02DA 558      : COMPLETION CODES:
02DA 559      :       NONE
02DA 560      :
02DA 561      : SIDE EFFECTS:
02DA 562      :       NONE
02DA 563      :
02DA 564      :--
```

```
00000000 566 .PSECT Y$LOWUSE ;This code can page
0000 567
0000 568 MMG$ALC_PGFLVBN::
51 00000024'FF41 D0 0000 569 MOVL @L^MMG$GL_PAGSWPVC[R1],R1 ;Get base address from index
53 50 FD 8F 53 DD 0008 570 PUSHL R3 ;Get a scratch register
14 A1 53 D7 000A 571 DECL R0 ;Bit # is base 0
05 1E 0015 572 10$: ASHL #-3,R0,R3 ;Byte # in bit map
04 00 B1 50 D1 0011 573 CMPL R3,PFL$$_BITMAPSIZE(R1) ;Legal page file vbn?
E4 0017 574 BGEQU 20$ ;Branch if illegal
001C 575 BBSC R0,@PFL$$_BITMAP(R1),30$ ;Free the page and branch
001C 576 20$:
0020 577 BUG_CHECK BADPAGFILD,FATAL ;Bad page file address specified
E4 52 F5 0025 578 30$:
52 18 A1 D1 002A 579 DECL PFL$$_FREPAGECNT(R1) ;Count another free page
EC 19 002E 580 INCL R0 ;Point to next VBN in file
53 8ED0 0030 581 SOBGTR R2,10$ ;Go back if not done yet
05 0033 582 DECL R2 ;Form a minus 1
0034 583 CMPL PFL$$_FREPAGECNT(R1),R2 ;Insure that counts still consistent
0034 584 BLSS 20$ ;Bugcheck if not
0034 585 POPL R3 ;Restore scratch
0034 586 RSB ; and return
0034 587
0034 588 .END
```

BADALLOC	0000010B	R	03
BUGS_BADPAGFILA	*****	X	03
BUGS_BADPAGFILD	*****	X	03
CRITMSG	0000015C	R	03
DYN\$C_PFL	= 00000023		
EXESGC_FLAGS	*****	X	03
EXESV_PGFLCRIT	*****	X	03
EXESV_PGFLFRAG	*****	X	03
FRAGMSG	0000010F	R	03
IOCSBROADCAST	*****	X	03
MMGSALC_PGFLVBN	00000000	RG	04
MMGSALLOCPAGFIL1	00000038	RG	03
MMGSALLOCPAGFIL2	000001AF	RG	03
MMGSALLOCSWPAREA	00000000	RG	03
MMGSALCPAGFIL	0000023A	RG	03
MMGSDEALLOCPAGFIL	00000243	RG	03
MMSGSL_MAXPFIDX	00000028	RG	02
MMSGSL_NULLPFL	00000000	RG	02
MMSGSL_PAGSWPVC	00000024	RG	02
MMSGSW_MINPFIDX	0000002C	RG	02
MPWSGW_MPWPFC	*****	X	03
OPASUCBO	*****	X	03
PFL\$B_ALLOCSIZ	= 00000022		
PFL\$B_FLAGS	= 00000023		
PFL\$C_LENGTH	= 00000024		
PFL\$L_BITMAP	= 00000000		
PFL\$L_BITMAPSIZ	= 00000014		
PFL\$L_FREPAGECNT	= 00000018		
PFL\$L_MAXVBN	= 0000001C		
PFL\$L_STARTBYTE	= 00000004		
PFL\$M_SWPFILFUL	= 00000004		
PFL\$V_INITED	= 00000000		
PFL\$V_SWPFILFUL	= 00000002		
PTESM_PGFLVB	= 003FFFFF		
PTESV_CHKPNL	= 00000015		
RSNS_SWPFIL	= 0000000A		
SCH\$RAVAIL	*****	X	03
SENDMSG	00000222	R	03
SGNSGW_SWPFILCT	0000002C	RG	02

-----  
! Psect synopsis !  
-----

## PSECT name

## Allocation

## PSECT No.

## Attributes

. ABS	00000000 ( 0.)	00 ( 0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
\$AB\$\$	00000000 ( 0.)	01 ( 1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
\$\$\$220	0000002E ( 46.)	02 ( 2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG
\$MMGCOD	000002DA ( 730.)	03 ( 3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
Y\$LOWUSE	00000034 ( 52.)	04 ( 4.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE



+-----+  
! Performance indicators !  
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	35	00:00:00.05	00:00:02.44
Command processing	117	00:00:00.55	00:00:03.46
Pass 1	187	00:00:03.93	00:00:11.36
Symbol table sort	0	00:00:00.35	00:00:01.55
Pass 2	114	00:00:01.42	00:00:05.32
Symbol table output	5	00:00:00.05	00:00:00.04
Psect synopsis output	2	00:00:00.04	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	462	00:00:06.39	00:00:24.32

The working set limit was 1350 pages.

22714 bytes (45 pages) of virtual memory were used to buffer the intermediate code.

There were 20 pages of symbol table space allocated to hold 267 non-local and 32 local symbols.

588 source lines were read in Pass 1, producing 19 object records in Pass 2.

12 pages of virtual memory were used to define 11 macros.

+-----+  
! Macro library statistics !  
+-----+

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	5
_\$255\$DUA28:[SYS.LIB]STARLET.MLB;2	3
TOTALS (all libraries)	8

319 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:PAGEFILE/OBJ=OBJ\$:PAGEFILE MSRC\$:PAGEFILE/UPDATE=(ENHS:PAGEFILE)+EXECMLS/LIB



0378 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800
801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900
901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000